Міністерство освіти і науки України
Національний університет водного господарства та природокористування
Кафедра гідроінформатики

01-02-161

Методичні вказівки

до практичних занять та самостійної роботи з навчальної дисципліни **«Використання вільного програмного забезпечення у водній інженерії»** для здобувачів вищої освіти першого (бакалаврського) рівня усіх освітньо-професійних програм спеціальностей НУВГП денної форми навчання

Схвалено науково-методичною радою НУВГП протокол № 2 від 26.03.2020 р.

Рівне – 2020

Методичні вказівки до практичних занять та самостійної роботи з навчальної дисципліни «Використання вільного програмного забезпечення у водній інженерії» для здобувачів вищої освіти першого (бакалаврського) рівня усіх освітньо-професійних програм спеціальностей НУВГП денної форми навчання [Електронне видання] / Новачок О. М. – Рівне : НУВГП, 2020. – 32 с.

Укладач: Новачок О. М., кандидат сільськогосподарських наук, доцент кафедри гідроінформатики.

Відповідальний за випуск: Клімов С. В., кандидат технічних наук, доцент, завідувач кафедри гідроінформатики.

Вчений секретар
науково-методичної ради Костюкова Т. А.

Зміст

Вступ

Методичні вказівки призначені для використання при вивченні дисципліни «Використання вільного програмного забезпечення у водній інженерії» студентами вільного вибору всіх спеціальностей денної форми навчання.

Метою навчальної дисципліни є оволодіння студентами сучасними методами та засобами прийняття інженерних рішень у водогосподарській галузі на засадах математичного моделювання та комп'ютерних технологій з використанням вільного програмного забезпечення; формування у студентів системного, аналітичного мислення для оцінки ситуацій, що виникають.

Основним завданням вивчення дисципліни «Використання вільного програмного забезпечення у водній інженерії» є:

- ознайомлення студентів з можливостями сучасного вільного програмного забезпечення з метою його використання у водогосподарській галузі;
- розкриття можливостей ефективного застосування інформаційних технологій у водній інженерії.

Після вивчення дисципліни студенти повинні ***знати***:

- особливості вільного програмного забезпечення;
- сучасні інформаційні технології у водному господарстві, вимоги до них;
- основні поняття та види геоінформаційних систем та технологій;
- системи для високоякісного оформлення документів.

Після вивчення дисципліни студенти повинні ***вміти***:

- використовувати сучасне вільне програмне забезпечення у водній інженерії, проводити необхідні оптимізаційні розрахунки, використовувати стандартне офісне програмне забезпечення;
- використовувати вільні геоінформаційні системи, джерела публічної інформації, вільне програмне забезпечення для вирішення водогосподарських проблем.

Проблема неліцензійного програмного забезпечення сьогодні стоїть особливо гостро. І якщо раніше тези про захист авторських прав на комп'ютерні програми закріплювались тільки на папері, то сьогодні вони знаходять своє втілення в реальному житті у вигляді перевірок контролюючими органами, штрафів, вилучення комп'ютерів і судових розглядів.

Цивільно-правова відповідальність:

Згідно ст. 432 Цивільного кодексу України правовласники програмного забезпечення мають право подати позов до судових органів у разі порушення їх авторського права. Можна виділити наступні можливі рішення суду (ч. 2 ст. 52 Закону України «Про авторське право і суміжні права»):

відшкодування моральної та матеріальної шкоди; стягнення з порушника доходів, отриманих в наслідок порушення; опублікування в ЗМІ інформації про порушення; виплата компенсації в розмірі **від 10 до 50 000 мінімальних зарплат** і ін.

1. Політика вільного ліцензування. Історія Linux: від ядра до дистрибутивів.

1.1 Історія виникнення вільного програмного забезпечення

1.1.1 Розробка програмного забезпечення як наукове дослідження

Особливість програмного забезпечення полягає в тому, що воно створюється в одній формі - у вигляді вихідного тексту (source code), а поширюється і використовується в іншій - у вигляді двійкової програми, машинних кодів, за якими неможливо однозначно відновити вихідний текст. Щоб змінювати програму, виправляти помилки або навіть просто точно встановити, що і як робить програма, необхідно маніпулювати її вихідним текстом.

Спочатку створення програмного забезпечення для комп'ютерів було в першу чергу академічним заняттям. Для фахівців в області комп'ютерної науки (computer science) кожна програма була результатом наукового дослідження, в деякому сенсі аналогічний публікації статті. Це означає, що вихідний текст програми був обов'язково доступний всьому науковому співтовариству, оскільки будь-який науковий результат повинен бути верифікованим, тобто підтверджуватися іншими дослідниками і бути відкритим для критики. Таким чином, процес розробки програмного забезпечення принципово більш схожий з науковим процесом: вчений брав існуючі програми, виправляв їх у відповідності зі своїми ідеями і публікував виправлені програми - новий результат.

Однак технологія виробництва комп'ютерів розвивалася не менш активно, ніж програмне забезпечення для них. У 1970-ті роки існувала величезна різноманітність архітектур обчислювальних машин, розрізнялися і продуктивністю, і ціною. Природно, для кожної архітектури доводилося розробляти окремий набір програмного забезпечення. З середини 1970-х в більшості американських університетів (де переважно і розвивалася комп'ютерна наука) для академічних розробок використовувалися комп'ютери архітектури PDP-10, що дозволило співробітникам різних університетів використовувати розробки один одного на своїх машинах. Співробітники лабораторії штучного інтелекту массачусетського технологічного інституту в кінці 1970-х розробили для PDP-10 власну операційну систему ITS (Incompatible Timesharing System, несумісна система з поділом часу) і дуже великий набір програм для неї. Вихідні тексти написаних в MTI програм були загальнодоступні, співробітники інших університетів користувалися їх вихідними текстами і надсилали їм виправлення. Все програмне забезпечення в цих лабораторіях було повністю

академічним, а серед вчених-розробників панував справжній дух співробітництва.

1.1.2 Програмне забезпечення як "патентований" продукт

В умовах величезного різноманіття архітектур програмне забезпечення складало невід'ємну частину самої машини. Виробники комп'ютерів поставляли їх разом з основним програмним забезпеченням - в крайньому разі, з операційною системою. Виробництво комп'ютерів було наукомістким, але в основі своїй комерційним підприємством.

У ситуації, коли програмне забезпечення є предметом продажу, на нього поширюються вже не тільки закони наукової розробки, а й властивості матеріальних предметів, якими можна торгувати, обмінюватися, право володіння і користування якими варто охороняти законодавчо. Так програмне забезпечення потрапило в розряд інтелектуальної власності, тобто вихідний текст програми став розглядатися як твір, об'єкт застосування авторського права. Щоб захистити свої інтереси, виробники комп'ютерів і програмного забезпечення використовують ліцензії - вид договору між власником авторських прав і користувачем (покупцем) програмного забезпечення. Подібні договори укладалися і з університетами - наприклад, університету передавалися вихідні тексти програм і право змінювати їх, але заборонялося поширювати ці тексти за межами університету. Подібні обмеження означали, що тексти відповідних програм не могли відкрито обговорюватися в суспільстві, тобто не існували для наукової розробки. Були у комп'ютерів і програмного забезпечення покупці і з поза університетських середовищ - наприклад, банки. Таким користувачам не настільки важливо отримати вихідні тексти програм, вони більше зацікавлені в програмному забезпеченні, як в готовому продукті і платять гроші за надійні і зручні програми.

У європейській культурі так довго вироблялися правила власності по відношенню до матеріальних предметів, що поширення цих прав на предмети нематеріальні - програмні продукти - виглядає справою природною і не викликає сумнівів. А приводів для сумнівів чимало. Головна відмінність програмного продукту від, припустимо, стільця - так зване беззбиткове копіювання. Якщо грабіжник відбирає у селянина стілець, відбувається злочин: селянин стільця позбавляється, терпить збитки. Якщо селянин віддає комусь стілець добровільно, він його також позбавляється, тому має право вимагати відшкодування збитків - наприклад, грошима. Для того, щоб шкоди у селянина не відбувалося, стілець потрібно відтворити: добути дощок, покликати столяра, червонодеревника і оплатити їх роботу, і один з двох одержаних предметів побуту віддати грабіжнику. В цьому випадку збиток - грошовий - терпить той, хто оплачує копіювання стільця. Цілком природно при цьому законодавчо забороняти нанесення збитку, тобто визнавати право розпоряджатися річчю тільки за однією людиною - за її господарем. Ніяких

додаткових механічних або юридичних пристосувань, що забороняють відтворювати стільці, при цьому не потрібно.

Інша річ - програмний продукт. Скільки б коштів не було вкладено в його розробку, процедура його копіювання (переписування з одного носія даних на інший) різко відрізняється від процедури відтворення стільця. Вона не вимагає участі жодного з авторів програми, ні, за великим рахунком, взагалі людини. Єдина видаткова стаття при цьому - ціна носія даних і амортизація копіювального пристрою. В результаті такого копіювання виходить два примірника програми, що створюють зручності вже для двох людей. Таким чином, якщо людина оцінює що принесені програмою зручності перевищують номінальну вартість носія даних, копіювання - благо. Якщо ж ставитися до програмного продукту, як до матеріальної речі, і закріплювати право її використання за якоюсь однією людиною, виникає безліч негараздів, кожну з яких доводиться вирішувати штучними, а часто і протиприродними методами.

Наприклад, доведеться вишукувати, яких збитків все-таки наноситься "господареві" програми при її беззбитковому копіюванні. Зазвичай при цьому фігурує поняття "упущена вигода", тобто той прибуток, яку господар міг би отримати, але не отримав через те, що продукт скопіювали. Пригадується історія 30-х років, коли радянський колгоспник вкрав мішок колгоспної пшениці, і був засуджений за розкрадання в великих розмірах: якщо, мовляв, цю пшеницю всю посіяти, та виростити, та зібрати, вийшло б кілька центнерів. Доводиться винаходити хитромудру апаратуру, що заважає копіювання або заподіює при цьому збиток. Доводиться вводити в законодавство особливу категорію прав, умовно назвемо її "патент", що обмежує зловживання - і свободу - всього людства на користь господаря патенту. Причому далеко не завжди господар патенту і автор винаходу - одна і та ж людина!

Термін "патентований програмний продукт" означає не наявність дійсного патенту на програму, а наявність у програми власника, який ставиться до неї як до матеріального об'єкту (в разі патентованих програм). Патентовані програми часто називають "пропрієтарними" (від англійського терміна"proprietary") або просто "комерційними" (що, строго кажучи, невірно, так як "робити комерцію" - тобто отримувати вигоду - можна різними способами, і багато успішних вільних проектів це підтверджують).

1.1.3 Поява вільного ПЗ

Комп'ютери розвивалися дуже швидко, і колишні цілком сучасні в 1970-і PDP-10 на початок 1980-х вже застаріли і значно відставали по продуктивності від більш сучасних машин. Однак ні для однієї з нових архітектур вже не було операційної системи та іншого програмного забезпечення, розробленого виключно в академічному середовищі. Університети змушені були купувати нові комп'ютери з новим програмним забезпеченням і виконувати умови ліцензії, що обмежує їх права на розробку

і поширення ПЗ. Таким чином, наукова модель розробки програмного забезпечення, характерна для UNIX, ставала незатребуваною.

В цей час в лабораторії штучного інтелекту MTI розроблялися так звані LISP-машини, які вміли на апаратному рівні інтерпретувати мову програмування, схожу на LISP - розвинену і перспективну мову програмування. На LISP була написана операційна система для таких машин і все програмне забезпечення для них. На початку 1980-х деякі співробітники лабораторії штучного інтелекту викупили у MTI права на LISP-машини і математичну систему MACSIMA і заснували власні комерційні компанії для подальших розробок у цій галузі. Дуже багато співробітників лабораторії перейшли працювати в ці компанії, після чого всі їх розробки вже ставали закритими для наукової спільноти. Нові LISP-машини поставлялись з ліцензіями, які забороняли користувачам модифікувати і поширювати вихідні тексти програм. Програми, які раніше для співробітників MTI були аналогом наукових публікацій, стали комерційним продуктом який комусь належав.

Одному зі співробітників, що залишився в лабораторії штучного інтелекту MTI, Річарду Столлману, такий стан справ здавався неприпустимим порушенням відкритого наукового процесу розробки програмного забезпечення. Він самотужки намагався в рамках колишньої академічної моделі розвивати LISP-машини і відкрито реалізовувати зміни, аналогічні зробленим в рамках закритої комерційної розробки, щоб LISP-машини MTI могли конкурувати з комерційними аналогами. Звичайно, ця спроба наздогнати активну комерційну розробку була приречена на невдачу.

Тоді в пошуках однодумців Річард Столлман створив некомерційну організацію Фонд вільного програмного забезпечення (Free Software Foundation, FSF). Своєю основною метою організатори Фонду бачать в збереженні програмного забезпечення, процес розробки якого завжди буде гарантовано відкритим, а вихідні тексти - завжди доступними. Більш масштабна мета Фонду - розробка операційної системи, що цілком складається з відкрито розроблюваного програмного забезпечення. Декларуючи таку мету, Столлман фактично хотів повернути, на його думку ідеальний стан, коли в MTI працювали у власній операційній системі для PDP-10.

Операційна система, що розроблялася в рамках Фонду, повинна була стати сумісною з операційною системою UNIX. Спочатку UNIX був розроблений в 1970-і роки Кеном Томпсоном і Деннісом Річі в лабораторії компанії AT&T і поширювався цією (а згодом - і іншими) компанією як комерційна операційна система. До початку 1980-х UNIX дуже широко використовувався, в тому числі і в академічному середовищі. Для цієї операційної системи існувало багато програм, які вільно поширювалися в науковому співтоваристві, тому хотілося, щоб ці програми працювали і в новій - вільній операційній системі. Ця майбутня операційна система отримала назву GNU 2.

1.1.4 Визначення вільного ПЗ

Для того щоб зберегти модель наукового співробітництва між розробниками, необхідно було зробити так, щоб вихідні тексти програм, написаних розробниками, залишалися доступними для читання і критики всьому науковому співтовариству. Річард Столлман сформулював поняття вільне програмне забезпечення, в якому відобразив принципи відкритої розробки програм в науковому співтоваристві, що склалося в американських університетах в 1970-і роки. Столлман явно ввів критерії вільного програмного забезпечення. Ці критерії обумовлюють ті права, які автор вільної програми передає будь-якому користувачеві:

• програму можна використовувати з будь-якою метою ("нульова свобода").

• можна вивчати, як програма працює і адаптувати її для своїх цілей ("перша свобода"). Умовою цього є доступність вихідного тексту програми.

• можна поширювати копії програми - в допомогу товаришеві ("друга свобода").

• програму можна покращувати і публікувати свою вдосконалену версію, з тим, щоб принести користь всьому співтовариству ("третя свобода"). Умовою цього є доступність вихідного тексту програми.

Тільки програма, яка задовольняє всім принципам може вважатися вільною, тобто гарантовано відкритою і доступною для наукового співтовариства. Потрібно підкреслити, що ці принципи обумовлюють тільки доступність програм для загального використання, критики і поліпшення, але ніяк не обмовляють пов'язані з поширенням програм грошові відносини, в тому числі не припускають і безоплатності. В англомовних текстах тут часто виникає плутанина, оскільки слово "free" означає не тільки "вільне", але і "безкоштовне", і воно часто використовується по відношенню до програмного забезпечення, яке розповсюджується без справляння плати за використання, але яке при цьому абсолютно недоступне для зміни співтовариством, просто тому, що його вихідні тексти не опубліковані. Таке безкоштовне ПО зовсім не є вільним. Навпаки, вільне ПЗ цілком можна поширювати, стягуючи при цьому плату, однак дотримуючись при цьому критеріїв свободи: кожному користувачеві надається право отримати вихідні тексти програм, змінювати їх і поширювати далі. Будь-яке програмне забезпечення, користувачам якого не надається такого права, є невільним.

1.1.5 Громадська ліцензія GNU

Задекларувавши критерії вільного ПЗ, члени Фонду вільного ПЗ стали поширювати свої програми відповідно до цих принципів, ніяк не оформлюючи це документально. Інакше кажучи, спочатку вільні програми поширювалися взагалі без ліцензії. Однак прецедент, який стався з самим Річардом Столлманом, переконав його в тому, що документальне оформлення необхідне для вільного ПЗ.

Річард Столлман займався розробкою текстового редактора Emacs на основі вихідних текстів Джеймса Гослінга (який згодом став автором відомого сьогодні продукту Java). Тоді Гослінг вільно роздавав свої вихідні тексти всім зацікавленим. Однак потім Гослінг продав права на поширення Emacs компанії UniPress, і компанія попросила Столлмана припинити поширення його версії Emacs, так як права належать їм. Цей інцидент змусив Столлмана переписати заново ті частини вихідного тексту Emacs, які тепер належали UniPress, після чого він розробив власну ліцензію на програмне забезпечення.

Ліцензія, сформульована Столлманом, повинна була працювати так само, як і ліцензії на комерційне програмне забезпечення: це типовий договір автора програми (власника авторських прав) з користувачем, в якому автор обумовлює права користувача по відношенню до програми. На відміну від комерційної ліцензії, в ліцензії Столлмана обумовлюються ті права, які користувач отримує по відношенню до вільної програми: отримувати вихідні тексти програм, змінювати їх, поширювати змінені і незмінені версії (див. перелічені вище критерії вільного ПЗ). Крім того, в цій ліцензії обмовляється принципова для Столлмана умова поширення вільного ПЗ: жоден користувач не має права, зробивши модифіковану версію вільної програми, поширювати її, не дотримуючись всіх принципів вільного ПЗ, обмежуючи тим самим права інших користувачів по відношенню до програми. Інакше кажучи, **не можна модифікацію вільної програми зробити невільною.**

Ліцензія, що містить таку умову, отримала назву "**copyleft**". Тут гра слів: англійською авторське право називається "copyright", буквально "копіювати-право", а "copyleft", відповідно, "копіювати-ліво". Дійсно, умова "copyleft" прямо протилежна за змістом авторського права: авторське право покликане обмежити користувача в копіюванні та розповсюдженні копій продукту, а "авторське ліво", навпаки, суворо забороняє його обмежувати. Згодом ліцензія Столлмана отримала назву "Громадська ліцензія GNU" (GPL, General Public License).

В даний час крім GPL відомі й інші ліцензії, під якими може поширюватися вільне ПЗ. Найпоширеніша з таких ліцензій - BSD License. Ліцензія BSD відрізняється від GPL головним чином тим, що в ній відсутня умова "copyleft", тобто на підставі вільного ПЗ, розповсюджуваного під цією ліцензією, можна виробляти невільні модифікації. Однак ліцензія BSD та інші ліцензії залишатимуться ліцензіями на вільне програмне забезпечення до тих пір, поки вони відповідають умовам, обговореним принципами вільного ПЗ, оголошеними Фондом.

1.1.6 Спільнота розробників і користувачів

Головною умовою існування вільного ПЗ є не ліцензія, а люди, які готові ділитися текстами своїх програм і вдосконалювати тексти чужих. Вільне ПЗ успадкувало модель відкритої наукової розробки, а разом з нею - і специфічну організацію спільноти розробників і користувачів, в деяких відносинах

нагадує академічне співтовариство. Щоб краще продемонструвати специфіку цієї спільноти, порівняємо соціальні відносини, які супроводжують використання вільного та патентованого ПЗ.

У будь-якого користувача програмного забезпечення неодмінно виникають питання, коли він намагається застосувати його для вирішення своїх завдань. Традиційна комерційна модель розробки і використання програмного забезпечення заснована на тому, що вихідні тексти програм є комерційною таємницею виробника, а користувач отримує готовий продукт - скомпільовану програму. Така програма є невільною. Користувач невільної (патентованої) програми платить за неї виробнику, який натомість надає йому деякі гарантії, одна з яких - відповідати на питання про роботу програми. Спеціально для цього виробник організовує службу підтримки, яка по телефону і по електронній пошті відповідає на запитання користувачів.

Користувач вільно розповсюджуваної програми не отримує разом з нею ніяких гарантій: автор зробив її вихідний текст відкритим для суспільства, але при цьому не брав на себе зобов'язань пояснювати всім, як працює програма. Тому отримати відповідь на своє питання користувач може з двох джерел: з документації, а якщо її недостатньо - від більш досвідчених користувачів. Добре, якщо такі користувачі є серед знайомих, а якщо ні? У цьому випадку їх завжди можна знайти в списку розсилки в Internet, присвяченому даній програмі.

Лист, який прийшов на електронну адресу списку розсилки, буде відправлено всім користувачам списку, будь-який з них може відповісти на нього, і відповідь також отримають всі... Так організується щось на зразок віртуальної спільної кімнати для розмов. В даний час склалося неписане правило, що для кожної вільно розповсюджуваної програми існує окремий список розсилки. Знайти адресу цього списку і підписатися на нього можна в Internet (зазвичай на сайті, присвяченому цій програмі). Будь-який користувач вільної програми може направити своє питання в список розсилки. Списки розсилки читають розробники програми і її активні користувачі, і зазвичай серед них знаходиться той, хто відповість на питання. Так виходить, що користувачі вільних програм, за відсутності централізованої служби підтримки, організовуються в співтовариство для взаємодопомоги.

У користувачів програм знову і знову виникають одні і ті ж питання і складності. Постійним читачам списків розсилки це особливо очевидно, оскільки їм доводиться на ці питання відповідати не по одному разу. У таких ситуаціях у них нерідко виникає ініціатива записати відповіді на найпоширеніші питання і відкрити їх для загального огляду. Так до вільної програми з'являється нова документація в жанрі FAQ (Frequently Asked Questions, питання які часто задають), що представляє собою список питань з відповідями. Користувачі патентованих програм теж ставлять одні і ті ж питання, тільки не в списку розсилки, а службі підтримки, в результаті так

само з'являється документація типу FAQ, яка чомусь рідко виходить за межі внутрішнього користування виробника програми.

У будь-якій програмі неодмінно є помилки (bugs). Виробник патентованої програми оплачує роботу відділу контролю якості, який займається пошуком помилок. Тим не менш, деякі помилки цей відділ пропускає, і вони досягають користувача. Користувач невільної програми, зіткнувшись з помилкою, не може виявити її причину (оскільки йому недоступні вихідні тексти програми), але, швидше за все, здатний описати помилку і умови, в яких вона виникає. Він може повідомити про помилку виробникові програми (зазвичай за допомогою звернення все в ту ж службу підтримки), і якщо там вирішать, що помилка дійсно в програмі, а не в роботі користувача, про неї буде повідомлено розробникам. В результаті користувач може очікувати, що в наступній версії програми помилка буде виправлена.

У вільно розповсюджуваної програми зазвичай немає оплачуваного відділу контролю якості. Значить, користувач може зіткнутися з ще більшою кількістю помилок, ніж в патентованої програмі. Тим актуальніше для нього можливість повідомити про помилку розробникам програми. Раніше в документації, яка супроводжує програму було прийнято вказувати електронну адресу, за якою розробники отримували повідомлення про помилки, bug report. Деякі вводили стереотипну форму для таких повідомлень, щоб полегшити і автоматизувати їх обробку. Уже це вимагає істотно вищої зв'язності спільноти в усьому світі, істотно більшої, ніж достатньо для закритої розробки.

Розробники та контролери-випробувачі патентованого продукту можуть ходити на службу в один і той же офіс, і там обмінюватися інформацією або витрачати певну частку робочого часу на складання і аналіз строгих звітностей, що містять повідомлення про помилки і рапорти про усунення несправностей. Така організація праці ефективна, якщо коло розробників невелике, а ввести загальну дисципліну відносно легко (наприклад, карати гривнею). Для відкритого проекту коло потенційних розробників не обмежене нічим, тому ефективність розробки набагато більше залежить від того, наскільки просто всім членам спільноти домовлятися між собою, а також від "свідомості" користувачів. Помітивши помилку в програмі, свідомий користувач не просто виправить її самостійно (що не завжди йому під силу), а оформить виразне повідомлення про помилку, а якщо виправлення готове, долучить до повідомлення і виправлення.

Простому і впорядкованому прийому і перенаправленню повідомлень про помилки служать системи відслідковування помилок (Bug Tracking System), найвідоміші з яких розроблені учасниками великих проектів для себе, а завдяки вільним ліцензіям використовуються повсюдно. Такі GNUTS (розроблена в GNU), Bugzilla (mozilla.org), JitterBug (проект Samba) або DebianBTS. Більш ранні версії орієнтуються на електронну пошту, пізніші включають в себе WWW-інтерфейс. Наприклад, за допомогою Bugzilla

організовується сайт в Internet, на якому користувач може заповнити форму повідомлення про помилку. Кожне повідомлення має свій номер, за яким можна потрапити на "персональну" сторінку даної помилки, де відображаються всі події з її приводу, від початкового повідомлення (відкриття) до виправлення (закриття). При кожній зміні в стані помилки Bugzilla розсилає всім зацікавленим особам (включаючи, природно, того хто повідомив про помилку і тих розробників які займаються даною програмою) листи по електронній пошті. Оскільки Bugzilla дозволяє залишати коментарі та прикладати файли, вона є повноцінним засобом для спілкування користувача з розробником з приводу помилки в програмі.

Принципова перевага користувача вільної програми полягає в тому, що у нього, на відміну від користувачів невільних програм, завжди є можливість заглянути в вихідні тексти. Звичайно, для багатьох користувачів вихідні тексти не більше зрозумілі, ніж виконувані файли. Однак при достатньому рівні знань в програмуванні користувач може встановити причину помилки в програмі і усунути її, виправивши відповідним чином вихідний текст. А якщо користувач зацікавлений у розвитку програми, то з його боку буде розумно не лише повідомити автора про помилку, а й надіслати йому свої виправлення до початкового тексту програми: автору залишиться тільки застосувати ці виправлення до тексту програми, якщо він знайде їх коректними і доречними. Пересилати автору виправлений текст програми цілком непрактично: він може бути дуже великим (десятки тисяч рядків), і автору буде нелегко розібратися, що ж змінено (а раптом зміни зроблені неграмотно?).

Щоб полегшити і автоматизувати процес внесення виправлень, Ларрі Уолл (Larry Wall) в 1984 році розробив утиліту patch ("латочка"), яка в формалізованому (але добре зрозумілою людині) вигляді описує операції редагування, які потрібно зробити, щоб отримати нову версію тексту. З появою цієї утиліти користувач, який знайшов і виправив помилку в програмі, міг надіслати автору невелику латочку, по якій автор міг зрозуміти, які зміни пропонуються, і автоматично "докласти" їх до свого початкового тексту. З появою patch набагато більше користувачів стало включатися в розробку програм з доступним вихідним текстом, чималу роль і тут зіграла мережа Usenet. Файли-латочки з виправленнями - обов'язковий атрибут сьогоднішньої розробки вільного програмного забезпечення.

Однак до чого обмежувати сферу застосування patch виправленням помилок? Якщо користувачеві програми не вистачає в ній якоїсь функції, то при належній кваліфікації він цілком може запрограмувати її сам і включити у вихідний текст програми. Природно, йому вигідно, щоб його доповнення потрапило в "головний", авторський варіант програми (його називають "upstream") і з'являлося в усіх наступних версіях: можна точно так само оформити його у вигляді patch і вислати автору. Цією можливості позбавлений користувач невільної програми, навіть якщо він досить

кваліфікований. Єдиний спосіб включити в програму потрібну йому функцію - звернутися до виробника (якщо програма патентована) з відповідним проханням, і сподіватися, що виробник вважатиме запропоновану функцію дійсно необхідною.

Чим більше у вільній програмі активних користувачів, готових вносити виправлення і доповнення і ділитися ними, тим надійніше працює і швидше розвивається програма. Причому така вільна модель відстеження і виправлення помилок для програми, у якій тисячі активних користувачів, може виявитися набагато ефективнішою, ніж у будь-якої патентованої програми: жодна компанія не може собі дозволити такий величезний штат співробітників у відділі контролю якості. Тому дійсно популярна вільна програма може виявитися набагато надійнішою патентованих аналогів.

Написати велику програму поодинці досить складно і навіть не завжди можливо, особливо якщо автор займається цим у вільний від роботи час. Більшість сучасних вільних програм пишеться групою розробників. Навіть якщо починала писати програму одна людина, і вона виявилася цікавою, до розробки можуть приєднатися активні користувачі. Щоб вони могли не тільки вносити окремі виправлення, а й взагалі всю розробку вести спільно, потрібні спеціальні інструменти. Крім patch, для організації спільної розробки ПО застосовуються системи контролю версій. Функції системи контролю версій полягають у тому, щоб організувати доступ до вихідних текстів програми для декількох розробників і зберігати історію всіх змін у вихідних текстах, дозволяючи об'єднувати і скасовувати зміни та ін. Найбільш рання вільна система контролю версій, RCS використовувалася ще на зорі вільного ПЗ абонентами мережі Usenet, потім на зміну їй прийшла більш розвинена CVS, але сьогодні і вона вважається багато в чому застарілою, і все частіше замінюється Subversion, Arch і іншими. До слова, названі системи контролю версій сьогодні активно використовуються і розробниками патентованого ПЗ для організації спільної розробки.

Потрібно зауважити, що переваги вільної розробки для користувача не слід перебільшувати. Не всі вільні програми в рівній мірі доступні для зміни користувачам, і це абсолютно не пов'язано з ліцензією на їх поширення. Важливий фактор тут - обсяг програми: якщо в ній десятки тисяч рядків (як, наприклад, в OpenOffice.org), то навіть кваліфікованому користувачеві буде потрібно занадто багато часу, щоб розібратися, що до чого. А якщо при цьому ще немає розумної документації... Розраховувати ж на те, що розробники дадуть відповідь на всі зауваження і пропозиції користувача негайним виправленням програми теж не можна, оскільки вони не несуть перед користувачем ніяких зобов'язань за якість програми. В цьому відношенні користувач патентованої програми може бути навіть в кращому становищі.

Дуже багато властивостей спільноти розробників і користувачів вільного програмного забезпечення є наслідком того, що всі його учасники зазвичай

займаються цією програмою з інтересу або тому, що ця програма - необхідний для них інструмент (наприклад, заробляння грошей). Час, витрачений ними на програму, не оплачується, тому немає ніякої надії, що обставини не зміняться і розробка не припиниться зовсім. Нерідкі випадки, коли розробка програми починається завдяки одному автору-ентузіасту, який приваблює багатьох до участі в розробці, а потім ентузіазм лідера гасне, а разом з ним загасає і розробка. На жаль, сьогодні існують тисячі вільних програм, так ніколи і не досягли версії 1.0, хоча "вигорання" лідерів і не єдина цього причина. Крім того, програма може бути необхідною, але "нецікавою", а тому не знайдеться і вільних розробників.

Місце вільних програм на сьогоднішньому ринку ПО дуже значне, і багато комерційних і державних підприємств використовують вільне ПЗ прямо або опосередковано. Власне, опосередковано всі користувачі Internet задіюють, наприклад, вільну програму Bind, яка надає службу DNS. Багато організацій, особливо що надають послуги через Internet, використовують вільний web-сервер Apache, від роботи якого безпосередньо залежить їх прибуток, не кажучи вже про сервери на платформі Linux. Вигода використання вільного ПЗ очевидна: за нього не доводиться платити, а якщо доводиться - воно коштує набагато дешевше патентованих аналогів. Головний недолік з точки зору комерційного користувача: розробники вільного програмного забезпечення не несуть ніяких зобов'язань за якість програми, крім моральних. Тому сьогодні великі корпорації, наприклад, Intel або IBM, знаходять за необхідне підтримувати проекти з розробки вільного програмного забезпечення, оплачуючи співробітників, які працюють в рамках цих проектів.

1.1.7 Історія Linux GNU без Linux

До 1990-го року в рамках проекту GNU були розроблені і постійно розвивалися вільні програми, що становлять основний інструментарій для розробки програм на мові Сі: текстовий редактор Emacs, компілятор мови Сі gcc, відладчик програм gdb, командна оболонка bash, бібліотека найважливіших функцій для програм на Сі libc. Всі ці програми були написані для операційних систем, схожих на UNIX. Це означає, що в них використовувався стандартний для UNIX механізм запиту ресурсів комп'ютера, необхідних програмі - системні виклики, які виконуються ядром операційної системи. За допомогою системних викликів програми отримують доступ до оперативної пам'яті, файлової системи, пристроїв введення та виведення. Завдяки тому що системні виклики виглядали більш-менш стандартно у всіх реалізаціях UNIX, програми GNU могли працювати (з мінімальними змінами або взагалі без змін) в будь-який UNIX-подібній операційній системі.

За допомогою наявних інструментів GNU можна було б писати програми на Сі, користуючись тільки вільними програмними продуктами, проте вільного UNIX-сумісного ядра, на основі якого могли б працювати всі ці

інструменти, не існувало. У такій ситуації розробники GNU змушені були використовувати одну з комерційних реалізацій UNIX, тобто змушені були наслідувати прийнятим в цих операційних системах архітектурним рішенням і технологіям і засновувати на них власні розробки. Ідеал Столлмана про наукову розробку ПО, вільної від рішень, рухомих комерційними цілями, був недоступний, поки в основі вільної розробки лежало комерційне UNIX-сумісне ядро, вихідні тексти якого залишалися таємницею для розробників.

1.1.8 Linux - ядро

У 1991 році Лінус Торвальдс, фінський студент, надзвичайно захопився ідеєю написати сумісне з UNIX ядро операційної системи для свого персонального комп'ютера з процесором архітектури Intel 80386, яка стала дуже поширеною. Прототипом для майбутнього ядра стала операційна система MINIX: сумісна з UNIX операційна система для персональних комп'ютерів, яка завантажувалася з дискет і вміщувалася в дуже обмежену в ті часи пам'ять персонального комп'ютера. MINIX був створений Енді Танненбаумом, як навчальна операційна система, яка демонструє архітектуру і можливості UNIX, але є непридатною для повноцінної роботи з точки зору програміста. Крім того, MINIX можна було використовувати тільки в некомерційних цілях. Саме повноцінне ядро для свого ПК і хотів зробити Лінус Торвальдс. Назву для свого ядра він створив з власного імені, замінивши останню букву і зробивши його схожим на анаграму слова UNIX.

Сумісність з UNIX в цей момент означала, що операційна система повинна підтримувати стандарт POSIX. POSIX - це функціональна модель сумісної з UNIX операційної системи, в якій описано, як повинна вести себе система в тій чи іншій ситуації, але не наводиться жодних вказівок, як це слід реалізувати програмними засобами. POSIX описував ті властивості UNIX-сумісних систем, які були загальними для різних реалізацій UNIX на момент створення цього стандарту. Зокрема, в POSIX описані системні виклики, які повинна обробляти операційна система, сумісна з цим стандартом.

Найважливішу роль в розвитку Linux зіграли глобальні комп'ютерні мережі Usenet і Internet. На самих ранніх стадіях автор Linux обговорював свою роботу і труднощі які виникають з іншими розробниками в телеконференції comp.os.minix в мережі Usenet, присвяченій операційній системі MINIX. Ключовим рішенням Лінуса стала публікація вихідних текстів ще "сирої" першої версії ядра під вільною ліцензією GPL. Завдяки цьому і все більшому поширенню мережі Internet дуже багато людей отримали можливість самостійно компілювати і тестувати це ядро, брати участь в обговоренні і виправленні помилок, і надсилати виправлення і доповнення до вихідних текстів Лінуса. Тепер над ядром працювала вже не одна людина, і розробка пішла швидше і ефективніше.

У 1992 році версія ядра Linux досягла 0.95, а в 1994 році вийшла версія 1.0. Це означало, що розробники, нарешті, визнали ядро в цілому закінченим і всі помилки (теоретично) - виправленими. В даний час розробка ядра Linux

- справа вже набагато більшої спільноти, ніж за часів до версії 1.0. Змінилася і роль самого Лінуса Торвальдса, який тепер не головний розробник, але головний авторитет: він традиційно оцінює вихідні тексти, які повинні бути включені в ядро і дає "добро" на їх включення. Проте, загальна модель вільної розробки співтовариством зберігається. В даний час паралельно завжди розробляється два варіанти ядра. Стабільна версія, яка вважається досить надійною і придатною для користувачів, отримує номер, який закінчується на парну цифру, наприклад, "2.4". Номер відповідної експериментальної версії ядра закінчується на непарну цифру - "2.5". Експериментальна версія адресована в першу чергу розробникам ядра, які тестують нові можливості.

1.1.9 GNU і Linux

Однак, як не можна зробити операційну систему без ядра, так і ядро буде марне без утиліт, які використовували б його можливості. Завдяки проекту GNU Лінус Торвальдс з самого початку мав можливість задіяти в Linux вільні утиліти: bash, компілятор gcc, tar, gzip і багато інших вже відомих і широко використовуваних додатків, які могли працювати з його UNIX-сумісним ядром. Так Linux відразу потрапив в хороше оточення і в поєднанні з утилітами GNU став дуже цікавим середовищем для розробників програмного забезпечення навіть на самій ранній стадії свого розвитку.

Принциповим кроком уперед було саме те, що з ядра Linux і утиліт і додатків GNU вперше виявилося можливим зробити повністю вільну операційну систему, тобто працювати з комп'ютером і, більш того, розробляти нове програмне забезпечення, користуючись тільки вільним програмним забезпеченням. Ідеал повністю некомерційної розробки Столлмана тепер міг бути реалізований в життя.

Однак поява теоретичної можливості втілення ідеалу не означало його негайної практичної реалізації. Сумісність Linux і утиліт GNU була обумовлена тим, що і те, і інше писалося з орієнтацією на одні й ті ж стандарти і практику. Однак в рамках цієї практики (безліч різних UNIX-систем) залишався великий простір для несумісності і різних рішень. Тому на початковому етапі розробки ядра кожен розроблений під Linux додаток GNU був для Лінуса черговим досягненням: першими стали bash і gcc. Таким чином, поєднання GNU і Linux дало можливість створити вільну операційну систему, але саме по собі ще не становило такої системи, тому що Linux і різні утиліти GNU залишалися розрізненими програмними продуктами, які писали різні люди, не завжди беручи до уваги те, що роблять інші. Основна ж якість системи - узгодженість її компонентів.

1.1.10 Виникнення дистрибутивів

Після певного періоду розробки, під Linux вже стабільно працював ряд найважливіших утиліт GNU. Скомпільоване ядро Linux з невеликим комплектом скомпільованих вже в Linux утиліт GNU становило набір інструментів для розробника програмного забезпечення, який бажав

використовувати вільну операційну систему на своєму персональному комп'ютері. У такому вигляді Linux вже не тільки годився для розробки, а й був операційною системою, в якій можна було виконувати якісь прикладні завдання. Звичайно, перше, чим можна було займатися в Linux - писати програми на Сі.

Спочатку, щоб отримати комп'ютер з працюючою системою Linux, розробники користувалися спеціальними комплектами дискет зі скомпільованим ядром Linux і утилітами: з цих дискет можна було завантажити Linux і працювати. Однак це не дуже зручно, коли потрібно працювати в Linux постійно, та й обсяг дискет накладав суттєві обмеження на подальше розширення системи і включення нових утиліт.

Коли завдання отримати комп'ютер з постійно діючою на ньому системою Linux стала затребуваною і досить поширеною, розробники в гельсінкському і техаському університетах стали створювати власні набори дискет, з яких скомпільоване ядро і основні утиліти можна було записати на жорсткий диск, після чого завантажувати операційну систему прямо з нього. Ці набори дискет - перші прототипи сучасних дистрибутивів Linux - комплекти програмного забезпечення, на основі яких можна отримати працюючу операційну систему на своєму комп'ютері. Потрібно відзначити, що в дистрибутив Linux з самого початку входили програмні продукти GNU. Насправді, кожен раз, коли згадується "операційна система Linux", мається на увазі "ядро Linux і утиліти GNU". Фонд вільного ПЗ навіть рекомендує називати це операційною системою GNU/Linux.

Однак скопіювати всі потрібні програми на жорсткий диск ще недостатньо, щоб отримати відповідне для потреб користувача операційне середовище (нехай навіть це дуже професійний користувач). Тому перші набори дискет можна тільки умовно назвати збірками. Щоб отримати працюючу операційну систему, потрібні спеціальні засоби установки і настройки програмного забезпечення. Саме наявність таких засобів і відрізняє сучасні дистрибутиви Linux. Іншою найважливішою задачею дистрибутива є регулярне оновлення. Програмне забезпечення, особливо вільне, - одна з галузей що швидко розвиваються, тому мало один раз встановити Linux, потрібно ще регулярно оновлювати його. Першим дистрибутивом в сучасному розумінні, який отримав широке поширення, став Slackware, створений Патріком Фолькердінгом (до речі, цей дистрибутив зберігся і до наших днів). Він був широко відомий користувачам Linux вже до 1994 року.

Незважаючи на те, що з появою перших дистрибутивів установка Linux вже не вимагала самостійної компіляції всіх програм з вихідних текстів, використання Linux залишалося долею розробників: користувач цієї операційної системи в той період її розвитку міг займатися майже виключно програмуванням. В крайньому разі, щоб вирішувати в ній інші повсякденні прикладні завдання (наприклад, читання електронної пошти, написання

статей тощо), він повинен був спочатку деякий час позайматися програмуванням і навіть розробкою самої системи Linux, щоб створити для себе відповідні прикладні програми або змусити їх працювати в Linux.

Однак розробники - теж люди, які пишуть і електронні листи, і статті, і навіть малюють картинки. Все програмне забезпечення для Linux було відкритим, тому незабаром стало з'являтися все більше прикладних програм для Linux, які використовувалися все більш широкою спільнотою, тому програми ставали надійнішими і отримували нову функціональність. Зрештою виникає ідея, що з Linux та GNU-додатків для Linux цілеспрямованими зусиллями невеликої групи розробників можна робити цілісні операційні системи, які підходять для дуже широкого кола користувачів, і продавати ці системи користувачам за гроші, як аналог і альтернативу існуючим комерційним операційним системам.

Вигода операційної системи, що цілком складається з вільного програмного забезпечення, очевидна - збирачі цієї системи не повинні нікому платити за програми, які в неї входять. Більш того, подальша розробка та оновлення наявних програм ведеться співтовариством розробників також абсолютно безкоштовно, не потрібно платити співробітникам, які займалися б цим. У підсумку витрати фірми, яка щороку збирає дистрибутив Linux для користувача, обмежуються оплатою програмістів, інтегруючих розрізнені додатки в систему і які пишуть програми для стандартизації процедур установки і настройки системи, щоб полегшити ці завдання непідготовленому користувачеві, а також витратами на саме видання отриманого дистрибутива. Для кінцевого покупця це означає принципове зниження ціни на операційну систему.

Першою успішною компанією, що працює за такою схемою, стала RedHat, що з'явилася в 1995 році. RedHat адресувала свої розробки не тільки професійним програмістам, але і звичайним користувачам і системним адміністраторам, для яких комп'ютер - в першу чергу офісне робоче місце або робочий сервер. Орієнтуючись на вже існуючі на ринку пропозиції для такого класу користувачів, фахівці RedHat завжди приділяли велику увагу розробці додатків з графічним інтерфейсом для виконання типових завдань з налаштування та адміністрування системи. Бізнес RedHat розвивався досить успішно, в 1999 році ця компанія акціонувалася - відразу після випуску акції росли в ціні дуже енергійно, проте потім ажіотаж спав. В даний час частка RedHat на ринку серверів і робочих станцій Linux дуже велика. Завдяки RedHat в співтоваристві користувачів Linux широке поширення отримав формат пакетів RPM.

Практично одночасно з RedHat з'явився проект Debian. Його завдання були приблизно тими ж - створити цілісний дистрибутив Linux і вільного програмного забезпечення GNU, однак цей проект був задуманий, як принципово некомерційний, що проводиться в життя спільнотою розробників, норми взаємодії в якому повністю відповідали б ідеалам

вільного ПЗ. Спільнота розробників Debian - міжнародна, учасники якого взаємодіють через Internet, а норми взаємодії між ними визначаються спеціальними документами -полісі (policy).

Спільнота розробників не одержує ніякого прибутку від продажу Debian - його версії поширюються вільно, доступні в Internet, можуть поширюватися і на твердих носіях (CD, DVD), але і в цьому випадку їх ціна рідко сильно перевищує вартість носія і націнку, окупає витрати на видання. Спочатку розробка Debian спонсорувалася Фондом вільного програмного забезпечення. Адресатами дистрибутивів Debian завжди в першу чергу були професійні користувачі, так чи інакше пов'язані з академічною розробкою програмного забезпечення, які готові читати документацію та власноруч організувати потрібний профіль системи. Орієнтація на таку аудиторію визначила деякі тенденції розвитку Debian: в ньому ніколи не було великої кількості "простих" графічних засобів настройки середовища, всіляких майстрів, проте завжди приділялося багато уваги засобам послідовної та єдиної інтеграції програмного забезпечення в єдину систему. Саме в Debian з'явився менеджер пакетів (APT). В даний час Debian - найпопулярніший дистрибутив Linux серед професіоналів в області інформаційних технологій.

Завжди, коли вільне програмне забезпечення виявляється затребуваним, негайно виникає безліч альтернативних рішень - так сталося і зі збірками Linux. Після 1995 року виникло (і продовжує виникати) величезна кількість комерційних компаній і вільних громад, які ставлять своїм завданням підготовку і випуск дистрибутивів Linux. У кожного з них - свої особливості, своя цільова аудиторія, свої пріоритети. До теперішнього часу на ринку дистрибутивів виділилося кілька лідерів, які пропонують більш-менш універсальні рішення і найбільш широко відомі. Крім уже названих RedHat і Debian слід назвати в ряду дистрибутивів , орієнтованих на рядового користувача, німецький SuSE і французький Mandrake (зараз - Mandriva), серед адресованих фахівцям - Gentoo. Але крім "великих" гравців на ринку дистрибутивів є набагато більша кількість менш поширених дистрибутивів. Тепер перед користувачем, які бажають встановити Linux, постає питання вибору дистрибутива. Критерії вибору - завдання, які передбачається вирішувати за допомогою Linux, рівень підготовки користувача, технології і майбутні контакти з тим співтовариством, яке займається розробкою дистрибутива.

2. Програмний продукт LibreOffice, як вільна альтернатива до Microsoft Office.

Історія розвитку, склад пакету, основні відмінності між LibreOffice та Microsoft Office. Застосування у водній інженерії.

LibreOffice - це безкоштовний офісний пакет із відкритим кодом, проект The Document Foundation. Він був відокремлений у 2010 році від OpenOffice.org, який був відкритою версією попередньої StarOffice. Набір

LibreOffice включає програми для обробки тексту, створення та редагування електронних таблиць, слайд-шоу, діаграм та малюнків, роботи з базами даних та складання математичних формул. Він доступний на 115 мовах.

Як власний формат файлів для збереження документів для всіх своїх додатків, LibreOffice використовує відкритий формат документа для офісних програм (ODF) або OpenDocument, міжнародний стандарт, розроблений спільно Міжнародною організацією зі стандартизації (ISO) та Міжнародною електротехнічною комісією (IEC). LibreOffice також підтримує формати файлів більшості інших великих офісних пакетів, включаючи Microsoft Office, за допомогою різних фільтрів імпорту та експорту.

LibreOffice доступний для різних обчислювальних платформ, включаючи Microsoft Windows, macOS та Linux, Android та iOS, а також у формі веб-офісного пакету LibreOffice Online. Це офіційний набір за замовчуванням найпопулярніших дистрибутивів Linux. Це найбільш активно розроблюваний вільний офіс з відкритим кодом, який приблизно в 50 разів перевищує активність розвитку Apache OpenOffice, іншого головного нащадка OpenOffice.org.

Проект був оголошений, а бета-версія була випущена 28 вересня 2010 року. З січня 2011 року (перший стабільний реліз) і жовтня 2011 року LibreOffice завантажувались приблизно в 7,5 мільйонів разів. Проект має 120 мільйонів унікальних адрес для завантаження з травня 2011 по травень 2015 року, за винятком дистрибутивів Linux, при цьому 55 мільйонів - з травня 2014 року по травень 2015 року.

LibreOffice Writer - текстовий процесор із подібною функціональністю та підтримкою файлів для Microsoft Word або WordPerfect. Він має широкі можливості обробки тексту WYSIWYG, але також може бути використаний як основний редактор тексту.

Як і у всьому наборі LibreOffice, Writer можна використовувати на різних платформах, включаючи Linux, FreeBSD, Mac OS X та Microsoft Windows.

LibreOffice Calc Програма з електронними таблицями, схожа на Microsoft Excel або Lotus 1-2-3. Він має декілька унікальних особливостей, включаючи систему, яка автоматично визначає серію графіків на основі інформації, доступної користувачеві.

LibreOffice Impress Презентаційна програма, що нагадує Microsoft PowerPoint. Презентації можна експортувати у вигляді файлів SWF, що дозволяє переглядати їх на будь-якому комп’ютері зі встановленим Adobe Flash Player.

LibreOffice Draw Редактор векторної графіки та інструмент діаграм, подібний до Microsoft Visio та CorelDRAW. Він забезпечує з'єднувачі між формами, які доступні в різних стилях ліній і полегшують побудову креслень, таких як блок-схеми. Він також включає функції, схожі на програмне забезпечення для настільних видавців, таких як Scribus та

Microsoft Publisher. Він також може використовуватись як редактор файлів PDF.

LibreOffice Math Додаток, розроблений для створення та редагування математичних формул. У додатку використовується варіант XML для створення формул, визначений у специфікації OpenDocument. Ці формули можуть бути включені до інших документів у наборі LibreOffice, таких як ті, створені Writer або Calc, шляхом вставки формул у документ.

LibreOffice Base Система управління базами даних, подібна до Microsoft Access. LibreOffice Base дозволяє створювати та керувати базами даних, а також готувати форми та звіти, що забезпечують кінцевим користувачам простий доступ до даних. Як і Access, він може бути використаний для створення невеликих вбудованих баз даних, які зберігаються у файлах документів (використовуючи базу даних Java HSQLDB та Firebird на базі C ++), а для більш складних завдань він також може бути використаний як front-end для різних систем баз даних, включаючи бази даних Access (JET), джерела даних ODBC / JDBC та MySQL, MariaDB, PostgreSQL або Microsoft Access.

3. Вільні комп'ютерні математичні системи.

Maxima - вільна комп'ютерна алгебраїчна система. R - мова програмування і програмне середовище для статистичних обчислень, аналізу та представлення даних в графічному вигляді. Встановлення в операційній системі Ubuntu. Ознайомлення з основними можливостями. Застосування у водній інженерії.

Система аналітичних обчислень Maxima і (якщо необхідно) обчислювальне середовище Octave - хороший вибір для проведення будь-якої навчальної задачі або серйозного дослідження, де потрібна математика - від курсової роботи до наукової або інженерної розробки високого класу. За допомогою цих пакетів простіше готувати і виконувати завдання, влаштовувати демонстрації і набагато швидше вирішувати дослідницькі та інженерні завдання.

В даний час комп'ютерні програми цього класу (пропрієтарні - Maple, Mathematica, MATLAB, MathCad і ін., або з відкритим кодом) знаходять найширше застосування в наукових дослідженнях, стають одним з обов'язкових компонентів комп'ютерних технологій, що використовуються в освіті. Ці системи мають дружній інтерфейс, реалізують безліч стандартних і спеціальних математичних операцій, забезпечені потужними графічними засобами і володіють власними мовами програмування. Все це надає широкі можливості для ефективної роботи спеціалістів різних профілів, про що говорить активне застосування математичних пакетів в наукових дослідженнях і викладанні.

Для студентів системи компютерної математики (СКМ) зручний засіб розв'язання різноманітних задач, пов'язаних з символьними перетвореннями

(математичний аналіз, вища математика, лінійна алгебра і аналітична геометрія і т.п.), а також засіб вирішення завдань моделювання статичних (описуваних алгебраїчними рівняннями) і динамічних (описуваних диференціальними рівняннями) систем. Крім того, добротна СКМ - хороший засіб створення графічних ілюстрацій і документів, що містять математичні формули і викладки. Зараз для проведення розрахунків з довільних технічних дисциплін студентами-нематематиками широко використовується пакет MatCad, в основі якого лежить ядро Maple. При деякому навику і наявності документації зв'язка Maxima + TexMacs або ядро Maxima +інтерфейс wxMaxima цілком розумна заміна MathCad в Unix-середовищі. А наявність універсального інтерфейсу у вигляді TexMacs або Emacs дозволяє об'єднувати в одному документі розрахунки, виконані в Maxima, Octave, Axiom і т.п. Для науковців і інженерів СКМ незамінний засіб аналізу постановки різноманітних задач моделювання.

Під системами комп'ютерної математики розуміють програмне забезпечення, яке дозволяє не тільки виконувати числові розрахунки на комп'ютері, але й виконувати аналітичні (символьні) перетворення різних математичних і графічних об'єктів. Всі широко відомі математичні пакети: Maple, Matlab, Matematica , дозволяють проводити як символьні обчислення, так і використовувати числові методи. В даний час такі системи є одним з основних обчислювальних інструментів комп'ютерного моделювання в реальному часі і знаходять застосування в різних областях науки. Вони відкривають також нові можливості для викладання багатьох навчальних дисциплін, таких як алгебра і геометрія, фізика, інформатика, економіка, статистика, екологія. Застосування СКМ істотно підвищує продуктивність праці науковця, викладача вузу. Кінцевим продуктом дослідження виступають публікації, підготовка, поширення і використання яких в даний час вимагає кваліфікованого застосування комп'ютера. Це стосується редагування тексту, виготовлення графічних матеріалів, ведення бібліографії, розміщення електронних версій в Інтернеті, пошуку статей та їх перегляду.

Де-факто зараз стандартними системами підготовки науково-технічних публікацій є різні реалізації пакету TeX та текстовий редактор Word. Крім того, необхідні мінімальні знання про стандартні формати файлів, конвертори, програми і утиліти, які використовуються при підготовці публікацій.

R - мова програмування і середовище вільного програмного забезпечення для статистичних обчислень і графіки, що підтримуються R Foundation for Statistical Computing. Мова R широко використовується для статистики та інтелектуального аналізу даних, для розробки статистичного програмного забезпечення. Опитування, дослідження інтелектуального аналізу даних і дослідження літератури наукових баз даних показують значне збільшення популярності в останні роки. Станом на березень 2019 р. R посідає 14 місце в індексі TIOBE (міра популярності мов програмування).

Можливості R розширюються через створювані користувачем пакети, які дозволяють спеціалізовані статистичні методи, графічні пристрої, можливості імпорту та експорту, інструменти звітності. Ці пакети розробляються в основному в R, а іноді і в Java, C, C ++ і Fortran.

Основний набір пакетів включено при встановленні R, з більш ніж 15 000 додаткових пакетів (станом на вересень 2018), доступних у Всеохопній мережі архівів Comprehensive R Archive Network (CRAN), Bioconductor, Omegahat, GitHub та інших.

Сторінка "Перегляди завдань" на веб-сайті CRAN перелічує широкий спектр завдань для яких доступні пакети R (у таких галузях, як фінанси, генетика, високопродуктивні обчислення, машинне навчання, медичні зображення, соціальні науки та просторова статистика). R також був визначений FDA як придатний для інтерпретації даних клінічних досліджень.

Інші ресурси R-пакета включають Crantastic, сайт спільноти для оцінки та перегляду всіх CRAN-пакетів, R-Forge, центральну платформу для спільної розробки пакетів R, R-програм і проектів. У R-Forge також розміщено багато неопублікованих бета-пакунків, а також розробки версій пакетів CRAN.

Розробка **R** відбувалась під істотним впливом двох наявних мов програмування: мови програмування S з семантикою успадкованою від Scheme. R названа за першою літерою імен її засновників Роса Іхаки (Ross Ihaka) та Роберта Джентлмена (Robert Gentleman) працівників Оклендського Університету в Новій Зеландії.

Незважаючи на деякі принципові відмінності, більшість програм, написаних мовою програмування S запускаються в середовищі R. R розповсюджується безкоштовно за ліцензією GNU General Public License у вигляді вільнодоступого вихідного коду або відкомпільованих бінарних версій більшості операційних систем: Linux, FreeBSD, Microsoft Windows, Mac OS X, Solaris. R використовує текстовий інтерфейс, однак існують різні графічні інтерфейси ористувача. R має значні можливості для здійснення статистичних аналізів, включаючи лінійну і нелінійну регресію, класичні статистичні тести, аналіз часових рядів (серій), кластерний аналіз і багато іншого. R легко розширюється завдяки використанню додаткових функцій і пакетів доступних на сайті Comprehensive R Archive Network (CRAN). Більша частина стандартних функцій R, написана мовою R, однак існує можливість підключати код написаний C, C++, або Фортраном. Також за допомогою програмного коду на C, C++, Java , .NET або Python можна безпосередньо маніпулювати R об'єктами.

Завдяки своїй S спадщині, R має більш сильне об'єктно-орієнтоване програмування об'єктів , ніж більшість статистичних мов обчислень. Розширення R також полегшується завдяки її лексичній області видимості правил. Іншою силою R є статичні графіки, які можуть виробляти графіки типографської якості, в тому числі математичні символи. Динамічні та інтерактивні графіки доступні через додаткові пакети. R має Rd, свій власний

LaTeX формат документації, яка використовується для забезпечення вичерпної документації, як онлайн в різних форматах , так і в друкованому вигляді.

R належить до інтерпретованих мов програмування і для роботи використовується командний інтерпретатор.

R підтримує концепцію об'єктно-орієнтованого програмування (ООП) включаючи generic функції, результат виконання яких залежить від аргументів (типу об'єктів), що передаються generic функції. В мові програмування R всі змінні є об'єктами, кожен об'єкт належить до певного класу. При цьому R має дві класові моделі: S3 та S4. Перша була реалізована від початку існування R, друга була додана у версії 1.7.0 з пакетом methods. S3 не є справжньою класовою системою, класи S3-об'єкта визначаються простим атрибутом - вектором символьних рядків. При цьому, при виконанні generic функцій, таких як plot() чи summary(), диспетчер методів шукає в таблиці методів метод, який узгоджується з іменем першого аргумента.

Хоча R орієнтована на розв'язок і аналіз статистичних задач, вона може використовуватися для матричних обчисленб з порівняльною швидкодією до математичних пакетів GNU Octave або MATLAB. Створено багато пакетів для статистичних обчислень, біоінформатики, оптимізації тощо. Середовище R містить засоби для візуалізації результатів обчислень (двовимірні, тривимірні графіки, діаграми, гістограми, діаграми (схеми) Ганта тощо). Графічні можливості R дозволяють створювати високоякісні графіки з різними атрибутами, зокрема математичні формули і символи. Іншою особливістю є функція Sweave яка дозволяє інтеграцію і виконання коду R в документах написаних за допомогою LaTeX з метою створення динамічних звітів. R de-facto став стандартом у міжнародній спільноті спеціалістів в галузі статистики, і широко використовується в розробках статистичних програм та аналізі даних. Згідно щорічного опитуванню Rexer's Annual Data Miner Survey в 2010 році, більшість (43%) серед опитаних спеціалістів з аналізу даних використовують у своїй роботі середовище R.

Можливості R значно розширюються додатковими пакетами (бібліотеками). Пакети розробляються безпосередньо користувачами R. Існує понад 15000 пакетів, доступних на сайті Comprehensive R Archive Network (CRAN), Omegahat, Bioconductor, R-Forge. На сторінці "Task View" веб-сайту CRAN розміщено список напрямків (Фінанси, Генетика, Хеміометрія і Математична Фізика, Навколишнє середовище, Суспільні науки) в яких використовується R і для яких доступні пакети на сайті.

Для роботи з R існує кілька графічних інтерфейсів (GUI). Графічна оболонка RGui разом з командною оболонкою (терміналом) R Console входять до базового пакету R у версії для Windows.

RStudio — зручне кросплатформне середовище розробки з відкритим кодом (існує можливість запуску на віддаленому linux сервері).

RKward — розширюване середовище розробки IDE

RapidMiner і розширення RapidMiner R — середовище розробки для аналізу і обробки даних з використанням R.

WEKA Java Gui for R (JGR) — кросплатформний термінал і редактор R написаний на Java Deducer — графічний інтерфейс для аналізів даних з використанням системи меню (подібний до SPSS). Розроблений для використання разом з JGR та RGui.

Rattle GUI — кросплатформний графічний інтерфейс, розроблений для добування даних (збору та аналізу даних).

R Commander — кросплатформний GUI з системою меню і доступними додатковими плагінами (базується Tcl/Tk).

RExcel — додаток до Microsoft Excel, який дозволяє використовувати можливості R.

Sage — середовище для математичних розрахунків з використанням інтерфейсу веб-браузера, бібліотек R і підтримкою гру.

Red-R — інтерфейс для аналізу, що використовує R.

Tinn-R — графічний інтерфейс.

Текстові редактори та середовища розробки (IDE) з частковою підтримкою R: gedit, Bluefish, IDE Eclipse, Kate, Vim, Emacs (Emacs Speaks Statistics), Crimson Editor, ConTEXT, Tinn-R[15], Geany, jEdit, Syn, TextMate — The Missing Editor for Mac OS X, SciTE, WinEdt (R Package RWinEdt), WPE, notepad++ і SciViews.

R доступна для використання у мовах програмуваннях Python (за допомогою пакета RPy), Perl (за допомогою модуля Statistics::R) і Ruby (за допомогою RSRuby).

Деякі пропрієтарні програмні продукти призначені для аналізу статистичних даних (напр. SPSS, STATISTICA, SAS) мають розширення, розроблені для інтеграції у свої структури функціоналу R. Заснована 2007 року компанія Revolution Analytics розпочала комерційну підтримку версії R під назвою ParallelR, розробленої спеціально для кластерів робочих станцій. В 2011 з'явилася можливість зчитувати і записувати дані у формат файлів SAS за допомогою пропрієтарного Enterprise R.

4. Вільні геоінформаційні системи

Геоінформаці́йна систе́ма - сучасна комп'ютерна технологія, що дозволяє поєднати модельне зображення території (електронне відображення карт, схем, космо-, аерозображень земної поверхні) з інформацією табличного типу (різноманітні статистичні дані, списки, економічні показники тощо). Також, під геоінформаційною системою розуміють систему управління просторовими даними та асоційованими з ними атрибутами. Конкретніше, це комп'ютерна система, що забезпечує можливість використання, збереження, редагування, аналізу та відображення географічних даних. Геоінформаційні технології - технологічна основа створення географічних інформаційних систем, що дозволяють реалізувати

їхні функціональні можливості. Інформаційно-обчислювальна система, призначена для фіксації, збереження, модифікації, керування, аналізу і відображення усіх форм географічної інформації. ГІС використовується багатьма дослідниками в галузі вивчення проблем навколишнього середовища, для визначення різних показників на географічній сітці.

За територіальним поділом ГІС поділяються на глобальні ГІС, субконтинентальні ГІС, національні ГІС частіше мають статус державних, регіональних ГІС, субрегіональних ГІС та локальних або місцевих ГІС.

ГІС розрізняють за предметною областю інформаційного моделювання, наприклад, міські ГІС, або муніципальні ГІС, природоохоронні ГІС. Найпоширенішими ГІС - земельно-інформаційні системи. Проблема орієнтації ГІС визначається розв'язуваними задачами в ній, серед них інвентаризація ресурсів (в тому числі кадастр), аналіз, оцінка, моніторинг, управління і планування, підтримка прийняття рішень. Інтегровані ГІС, ІГІС (integrated GIS, IGIS) поєднують функціональні можливості ГІС і систем цифрової обробки зображень (даних дистанційного зондування) в єдиному інтегрованому середовищі.

Полімасштабні, або масштабно-незалежні ГІС засновані на множинних, або полімасштабних уявленнях просторових об'єктів, забезпечуючи графічне або картографічне відтворення даних на будь-якому з обраних рівнів масштабного ряду на основі єдиного набору даних з найбільшою просторовою роздільною здатністю.

Просторово-часові ГІС оперують просторово-часовими даними.

Реалізація геоінформаційних проектів, створення ГІС в широкому сенсі слова, включає етапи:

• передпроектних досліджень у тому числі вивчення вимог користувача і функціональних можливостей використовуваних програмних засобів ГІС, техніко-економічне обґрунтування, оцінку співвідношення «витрати / прибуток»;

• системне проектування ГІС, включаючи стадію пілот-проекту, розробку ГІС;

• тестування на невеликому територіальному фрагменті, або тестовій ділянці, прототипування, або створення дослідного зразка, або прототипу;

• впровадження ГІС;

• експлуатацію та використання.

Наукові, технічні, технологічні та прикладні аспекти проектування, створення та використання ГІС вивчаються геоінформатикою.

Особливості:

• візуалізація інформації у вигляді електронних карт.

• автоматична зміна зображеного образу об'єкта в залежності від зміни його характеристик.

• зміна масштабу та деталізація картографічної інформації.

Застосування ГІС є ефективним в різноманітних предметних областях, де важливі знання про взаємне розташування та форму об'єктів у просторі (екологія, сільське господарство, управління природними ресурсами, земельні та майнові кадастри, комунікації, містобудування та ландшафтне проектування).

Дані в ГІС поділяються на позиційні та атрибутивні.

Позиційні дані описують просторові характеристики різних об'єктів, таких як дороги, будівлі, водойми, лісові масиви. Реальні об'єкти можна розділити на дві абстрактні категорії: дискретні (будинки, територіальні зони) і неперервні (рельєф, рівень опадів, середньорічна температура).

Атрибутивна інформація

У ГІС до векторних об'єктів можуть бути прив'язані семантичні дані. Наприклад, на карті територіального зонування до просторових об'єктів, які становлять зони, може бути прив'язана характеристика типу зони. Структуру і типи даних визначає користувач. На основі атрибутивних значень, присвоєних векторним об'єктам на карті, може будуватися тематична карта, на якій ці значення позначені кольорами відповідно до шкали кольорів або різного роду штриховками чи крапом. Найчастіше атрибутивні дані зберігаються у таблицях реляційної бази даних та є прив'язаними до певних векторних об'єктів. У випадку використання растрового способу позиційна та атрибутивна інформація поєднуються — колір пікселя передає одночасно і розташування і характеристику.

Ба́зова ка́рта (англ. Base map) - карта, що містить основну (базову) топографічну інформацію в цифровому вигляді в одному чи кількох шарах. Використовується як стандартна структура, на яку накладаються додаткові конкретні дані та для контролю інших джерел просторових даних.

QGIS (раніше відомий як «Quantum GIS») — вільна крос-платформова геоінформаційна система (ГІС). QGIS є однією з найбільш функціональних і зручних настільних геоінформаційних систем що динамічно розвиваються.

Основним призначенням системи є обробка і аналіз просторових даних, підготовка різної картографічної продукції. Інтерфейс QGIS побудований на базі бібліотеки Qt. Пакет має гнучку систему розширень, які можна створювати на мовах C++ і Python. Підтримуються різноманітні векторні і растрові формати, включаючи ESRI Shapefile і GeoTIFF. GIS QGIS дозволяє користувачам створювати карти з безліччю шарів, використовуючи різні картографічні проекції. Карти можуть бути зібрані в різні формати і використовуватися для різних цілей. У системі QGIS карти можуть складатися з растрових або векторних шарів. Типовими для такого роду програмного забезпечення, векторні дані зберігаються як точка, лінія, полігон.

Різні види растрових зображень підтримуються і програмне забезпечення може виконувати геоприв'язку зображень.

QGIS забезпечує інтеграцію з іншими відкритими ГІС-пакетами, в тому числі PostGIS, GRASS і MapServer, щоб дати користувачам широкі функціональні можливості. Плагіни, написані на Python, C++, розширюють можливості QGIS. Є плагіни для геокодування за допомогою Google Геокодування API, виконання геообробки (fTools) схожими на стандартні інструменти ArcGIS, інтерфейс з PostgreSQL/PostGIS, SpatiaLite і MySQL баз даних, і використовування Mapnik як карту візуалізації.

Гарі Шерман почав розробку Quantum GIS на початку 2002 року, і це стало інкубатором проекту Open Source Geospatial Foundation в 2007 році. Версія 1.0 була випущена в січні 2009 року. QGIS широко використовує бібліотеки Qt. QGIS дозволяє інтеграцію плагінів, використовуючи C++ або Python. На додаток до Qt, необхідні залежності QGIS включають GEOS і SQLite. GDAL, GRASS GIS, PostGIS і PostgreSQL, так як вони забезпечують доступ до додаткових форматів даних.

QGIS працює під управлінням різних операційних систем, включаючи Mac OS X, Linux, UNIX та Windows. Для користувачів Mac, перевага QGIS над GRASS GIS у тому, що вона не вимагає віконної системи X11 для того, щоб переміщатися, а сам інтерфейс набагато чистіший і швидший.

QGIS може також використовуватися як графічний інтерфейс користувача у GRASS. QGIS має невеликий розмір файлу порівняно з комерційними ГІС і вимагає менше пам'яті і потужності процесора; отже, його можна використовувати на старих комп'ютерах або паралельно з іншими додатками, де потужності процесора можуть бути обмежені.

QGIS веде активна група добровольців-розробників, які регулярно випускає оновлення та виправлення. Станом на 2012 рік, розробники перевели QGIS на 48 мов і цей додаток використовується на міжнародному рівні в академічному або професійному середовищі.

Як вільне програмне забезпечення відповідно до ліцензії GNU GPL, QGIS може бути вільно змінена для виконання різних або більш спеціалізованих завдань.

Є два приклади: QGIS Браузер і QGIS серверних додатків, які використовують один і той же код для доступу до даних і візуалізації, але представляють різні інтерфейси. Також є безліч плагінів, що розширюють базову функціональність програмного забезпечення.

QGIS дозволяє використання DXF, шейп файли, покриття і персональні бази геоданих. MapInfo, PostGIS і ряд інших форматів підтримуються в QGIS. Веб-сервіси, в тому числі Web Map Service та Web Feature Service, також підтримуються, щоб дозволити використання даних із зовнішніх джерел.

Ряд державних і приватних організацій взяли QGIS. Серед інших, це програмне забезпечення є в Австрійському штаті Форарльберг, і в Швейцарії, в Кантоні Золотурн.

GRASS (англ. Geographic Resources Analysis Support System — Система Підтримки Аналізу Географічних Ресурсів) - це безплатна геоінформаційна

система (ГІС) з відкритим кодом, призначена для геомоделювання, управління просторовими растровими, векторними даними та комп'ютерної графіки, обробки супутникових знімків, створення карт, просторового моделювання і візуалізації. GRASS GIS містить понад 350 модулів для рендерингу карт; маніпулювання растровими і векторними даними, в тому числі векторними мережами; обробляти дані мультиспектральних зображень; а також створювати, управляти і зберігати просторові дані. Вона має ліцензію і випущена безкоштовно, з відкритим вихідним кодом під ліцензією GNU General Public License (GPL). GRASS GIS працює на декількох операційних системах, включаючи OS X , Windows і Linux. Користувачі можуть взаємодіяти з функціями програмного забезпечення за допомогою графічного інтерфейсу користувача (GUI) або шляхом підключення до GRASS за допомогою іншого програмного забезпечення, такого як QGIS. Вони можуть також взаємодіяти з модулями безпосередньо через зроблені на замовлення оболонки, які запускають додаток або шляхом виклику окремих модулів безпосередньо зі стандартної оболонки. Остання версія стабільний реліз (LTS) є GRASS GIS 7, який доступний з 2015 року. Команда розробників GRASS є багатонаціональною групою, яка складається з розробників у багатьох місцях. GRASS є одним з восьми вихідних програмних проектів в Geospatial Foundation Open Source.

GRASS підтримує обробку растрових і векторних даних в двох та трьох вимірах. Модель векторних даних ґрунтується на топології, що означає, що області визначаються границями та центроїдами; границі не можуть перекриватись на одному шарі. Такий підхід протилежний стандарту Simple Features консорціуму OpenGIS, який визначає вектори набагато вільніше, подібно до систем векторної графіки загального призначення.

GRASS розроблено як середовище, в якому виконуються різні інструменти, призначені для виконання специфічних для ГІС функцій. На відміну від звичайного прикладного програмного забезпечення, після запуску GRASS користувачу відображається модифікований командний процесор UNIX для виклику команд GRASS (також має назву модулів).

Більшість із модулів GRASS та можливостей системи доступні через графічний інтерфейс користувача (який реалізовано в модулі GRASS). В базову поставку GRASS включено приблизно 350 основних модулів, і понад 100 модулів доступні на сайті GRASS. Бібліотеки GRASS та основні модулі написано мовою програмування C; інші модулі написано на C, C++, Python, UNIX shell, Tcl та інших мовах програмування.

Модулі GRASS створювались відповідно до філософії UNIX, і, тому, можуть комбінуватись в скриптах для створення нових модулів, що розв'язують специфічні задачі користувачів. Існує модуль підтримки взаємодії з Quantum GIS (QGIS). Останні версії QGIS можуть виконуватись в середовищі GRASS, перетворюючи QGIS на дружній графічний інтерфейс,

більш схожий на графічні інтерфейси типових ГІС. Також існує проект, реалізації GRASS на платформі Java, відомий як JGRASS.

GRASS є однією із найстаріших ГІС і бере початок з розробок Армії США, що проводилися на початку 1980-х років. Розробка GRASS розпочалась в 1982 році, в ній брала участь велика кількість Федеральних агенцій США, навчальних закладів, приватних підприємств. Основні компоненти GRASS і координація зусиль учасників проекту GRASS знаходились у відомстві дослідницької лабораторії збройних сил США (USA-CERL). USA-CERL завершила свій останній випуск GRASS як версію 4.1 в 1992 році, і забезпечила п'ять оновлень і виправлень до цього випуску до 1995 року. USA-CERL також розробила основні компоненти GRASS.

Вихідний код проекту був відкритий в 1995 році під ліцензією GPL. У 2005 році випуск GRASS 6 додав підтримку нових 2D/3D топологічних даних та аналіз векторних мереж. Атрибути даних зберігаються в .dbf файлах або в основаних на SQL СУБД як, наприклад MySQL, PostgreSQL/PostGIS, і SQLite. Система може застосовуватись для візуалізації 3D векторної та воксельної графіки. GRASS підтримує широкий діапазон растрових і векторних форматів через використання бібліотеки GDAL/OGR. У випуску 7, що вийшов у лютому 2015, на зміну старому графічному інтерфейсу, написаному з використанням Tcl/Tk, прийшов новий мобільний інтерфейс на базі Cairo і wxPython. У інтерфейсі максимально спрощено виконання складних ГІС-операцій. Доданий новий Python API для доступу до функцій написаного на мові Сі ядра GRASS, що значно спрощує створення модулів GIS — Python. Усі модулі на мові bash перетворені в модулі на мові Python. Як драйвер БД за умовчанням замість DBF задіяний SQLite. Значно розширені можливості векторної бібліотеки і прискорена обробка векторних даних (наприклад, продуктивність деяких векторних модулів зросла в тисячу разів. Додана велика підбірка нових модулів для обробки і класифікації зображень, оцінки наростання біомаси і сумарного випаровування, визначення хмар на знімках, перевірки стану контрольних точок і таке інше. Додана можливість виведення в стандартних растрових форматах. Реалізовані модулі для роботи з тривимірними растровими даними.

5. Вільні текстові процесори

TEX - мова розмітки даних спеціального призначення, яка є основним ядром системи набору публікацій (комп'ютерної верстки); зокрема для набору математичних та інших технічних текстів. На базі TEX створено багато різноманітних пакетів для полегшення оформлення документів, найвідомішим серед них є LATEX.

Система TEX створена американським дослідником Дональдом Кнутом (англ. Donald Ervin Knuth, 10 січня 1938). Робота над TEX розпочалась в 1970 під час написання Д. Кнутом відомої праці «Мистецтво програмування».

Через рік після початку роботи над TEX, Д. Кнут був запрошений до Американського Математичного Товариства (AMS) прочитати лекцію на зібранні Товариства. Темою його доповіді були TEX (для набору), і Metafont (для розробки шрифтів, що використовуються в TEX). На той час TEX все ще був більше дослідницьким проектом, проте мав деякі особливості:

• він був призначений для використання безпосередньо авторами;

• він походив з академічного середовища, та розповсюджувався безкоштовно;

• був спроектований таким чином, щоб бути доступним на будь-якому комп'ютері і операційній системі, та видавати однакові результати роботи на різних системах.

З самого початку TEX став популярним серед математиків, фізиків, астрофізиків, астрономів, різних вчених-дослідників, яким бракувало необхідних символів на друкарських машинках, а якість документів, створених в інших системах, була незадовільною.

Ядро TEX — це інтерпретатор. Базові команди мови дуже низького рівня, але на їх основі можна створювати макрокоманди вищого рівня.

TEX істотно відрізняється від поширених сучасних систем підготовки документів. Вхідні файли мають вигляд звичайних текстових файлів, в тексті яких присутні спеціальні команди або макроси TEX.

Найбільшим дистрибутивом є кросплатформений TeXLive.

LaTeX (вимовляється «латéх») — мова розмітки даних та пакет макросів TeX для високоякісного оформлення документів. Вважається стандартом де-факто для підготовки математичних і технічних текстів для публікації в наукових виданнях.

Був створений Леслі Лампортом (англ. Leslie Lamport) на початку 1980-х років.

На відміну від текстових процесорів, особливу увагу в LaTeX приділено відокремленню змісту статті від оформлення. LaTeX пропонує засоби для підготовки структурованих документів - документів, автор яких має можливість основну свою увагу зосередити на змісті, а оформлення і решту рутинної роботи перекласти на програму. Як і у випадку TeX - вхідні файли LaTeX можна порівняти із програмами.

LyX - перший та єдиний візуальний редактор/процесор тексту для LaTeX. Головна ідея у LyX — це визначити що ви робите, а не як це зробити. Замість “Маєте те, що бачите,” “WYSIWYG” парадигма LyX — “What You See Is What You Mean” (“Маєте те, що мали на увазі”) або “WYSIWYM.” Ця потужна ідея значно спрощує механізм написання документів.

Призначений для користувачів, які вимагають високої якості та зручності при введенні тексту. LyX дає змогу автору зосередити увагу на змісті та логічній структурі документа. LyX зберігає документи у текстовому форматі подібному до XML. На момент компілювання документа LyX перетворює документ зі свого формату у формат LaTeX. Для використання програми не

потрібно знання LaTeX, хоча це знання дає змогу повніше використовувати програму.

Слід відмітити, що структурування документа повністю лягає на програму, а це нумерація, і слідкування за її правильністю по мірі змінювання і доповнення документа, таких елементів структури тексту, як частина, розділ, підрозділ, підпідрозділ, абзац, автор, дата, малюнки, таблиці, посилання та інше. Також програма автоматично, за бажанням користувача, створює переліки та сторінки переліків цих структурних елементів з автоматичною нумерацією. Також в програмі реалізовано простий механізм створення перехресних посилань та створення предметного та авторського покажчика (це може бути будь який елемент тексту). Це корисно у разі написання книжок та дисертацій, коли потрібно внести сторінки на яких розміщені таблиці, рисунки, або їх переліки. У цьому разі користувач звільняється від повторного введення цих елементів, все, що потрібно, вказувати на них під час створення тексту, а LyX має зручні інструменти для цього.

Рекомендована література

1. Сайт розробників операційної системи Ubuntu.
   URL: https://www.ubuntu.com/
2. Сайт розробників вільної комп'ютерної алгебраїчної системи Maxima.
   URL: http://maxima.sourceforge.net/
3. Сайт розробників мови програмування R, програмного середовища для статистичних обчислень, аналізу та представлення даних в графічному вигляді. URL: https://cran.r-project.org/
4. Сайт розробників програмного забезпечення вільної геоінформаційної системи QGIS та документація до неї.
   URL: http://www.qgis.org/uk/docs/index.html
5. Сайт розробників системи підтримки аналізу географічних ресурсів - Geographic Resources Analysis Support System (GRASS GIS)
   URL: https://grass.osgeo.org/
6. Сайт користувачів та розробників програмного забезпечення TeX та документація до нього. URL:http://tug.org/
7. Сайт розробників програмного забезпечення LyX та документація до нього. URL: http://www.lyx.org/